1 FAS Agent

High Level and Detailed Design Specification

Wireless NextGen

PPS ID: 1186 Feature ID: FAS Agent

Feature Activation System - Agent

Approvers:

TITLE	NAME	SIGNATURE & DATE
Product Management	Lee Rosenbaum	
FAS Engineering Lead	Ching Kung	
Program Manager	Carolyn Heide	
FAS Architect	Christian Rigg	

Revision History:

Originator:	**Specify Revisions Made to Form**	**Rev.**	**Date Released**
Himanshu Shekhar	Initial Document	0.1	XXXXXXX
Himanshu Shekhar	Changes after review	1.0	XXXXXXX

2 PURPOSE And Overview

The purpose of this document is to provide the high level and detailed design for the FAS-Agent component of the FAS system. FAS-Agent runs as a daemon process on the System Manager. It communicates with the FAS-SubAgent and FAS-Server over the FASP protocol to distribute and manage feature keys. SNMP and CLI are two user interfaces provided for access to the FAS-Agent.

3 Scope

This document provides the detailed design for Feature Activation System (FAS) Agent. In Total Control 2000 system (Wireless), the System Manager has the FAS Agent. This feature is required by the Wireless Access Gateway V.1 blade. This document specifies the design document of the System Manager to support FAS for Wireless Access Gateway V.1 blade. This document doesn't specify activation of features on System Manager and Shelf Controller in Total Control 2000 system. However, the specification mentioned in this document may be enhanced to support feature activations in System Manager and Shelf Controller in Total Control 2000 system.

The TC2000 system is the next generation carrier grade platform designed for the rapidly growing needs of the wire-line and wireless carriers. The new chassis design provides numerous opportunities to enhance the architecture of the wireless access platform and also presents significant migration opportunities.

4 DEFINITIONS

Feature Activation System (FAS) – Feature Activation System consists of having a feature key to communicate what features are permitted to be activated and having processing entities that act on the activation request. The feature activation process can be envisioned as being handled by a combination of processing entities each of which has certain responsibilities. A part of the Feature Activation System must be present in each Network Element in order to allow the customer to apply activation without reliance on any system external to the Network Element.

Feature Key Generator - This is the component that generates Feature Keys. It is operated by CommWorks. A database is needed for storing Feature Key specific information. There is no direct communications between the Feature Key Generator and the customer's network.

Feature Identifier – This is the identifier for a particular feature category, like QOS, IPSEC, etc.

CEM - This is the SNMP manager that communicates to the SNMP Agents such as SM, FAS-Server, FAS-Agents, etc.

FAS Server - This is the component of the Network Management Station that distributes Feature Keys to the Network Elements. There is one FAS Server per customer network management domain (FAS Domain). It can reside on the same machine as the CEM Server. The FAS Server software is distributed together with the CEM software package.

FAS Agent – This is the subsystem in the Network Element that receives and stores Feature Keys. It divides and distributes Feature Keys to the Application Modules. It resides in the Network Element Manager (System Manager in Total Control 2000 system) part of the NE.

FAS Sub-agent – This is the subsystem in the Application Module that receives Feature Keys from the FAS Agent. In Total Control 2000 system, AM-agents shall have FAS-Subagent.

Feature Key – This is a piece of data that is used by the FAS Server and the FAS Agent to activate a feature.

Feature Unit (FU) - One *Feature Unit* is the elementary unit (permission) of a feature. One Feature Unit is needed to activate one Feature Category on one FAS Sub-agent. The Feature Key includes an [1 to X] number of Feature Units (permissions).

Network Feature Key – Feature key(s) generated by the CommWorks Feature Key Generator in XML file format for the FAS Server. This is based on the serial number/host id of the machine where FAS Server is running. This is generated for customers.

Element Feature Key - Feature key(s) generated by the CommWorks Feature Key Generator or by FAS Server in XML file format. These are used by the FAS Agent (System Manager in Total Control 2000 system). This is based on the serial number of the control shelf in Total Control 2000 system. These keys are generated automatically by the FAS Server from the Network Feature Keys, or by CommWorks using the Feature Key Generator if the serial number of the control shelf is available.

System Manager (SM) - The System Manager provides the single point of network management for the Total Control 2000 system. It is responsible for storage and distribution of software loads and initial configuration for all the Total Control 2000 application blades. The System Manager also provides the SCB interconnects for supporting multi-shelf systems. SM has the FAS-Agent.

FAS Agent MIB – This is the feature activation system MIB on the FAS Agent. It has two tables –NE Feature Key Status Table and Blade Feature Key Status Table.

NE Feature Key Status Table - contains objects that are read-only. This table gives the overall picture of the total number of feature units available per feature units and their availability/allocation. This table is contained within FAS Agent MIB.

Blade Feature Key Status Table – contains objects that are read-only. This table gives the overall picture of feature allocation on each FAS sub-agent within FAS Agent control. This table provides information which blade has been activated which feature identifier, and how many feature units per feature identifier are in that blade. It also tells if the features are enabled in those blades.

FAS Sub-agent MIB – This is the feature activation system MIB for the FAS sub-agent. Each sub-agent blade owns this MIB. A module feature configuration table is in this MIB. This is a read-create MIB.

Feature Serial Number – This is a serial number embedded in the feature key that gets generated by the Feature Key Generator at CommWorks. The feature key serial number is maintained by CommWorks for each feature identifier issued. FAS-Server propagates the feature key serial number from Network Key to the Element Keys.

4.1 Acronyms and Abbreviations

AM	Application Module
CEM	Common

	Element Manager
CLI	Command Line Interface
CPS	CommWorks Professional Services
FAS	Feature Activation System
FASS	FAS Server
FASP	FAS Protocol
FFD	Feature Functional Document
GUI	Graphical User Interface
IPSec	Internet Protocol Security
MIB	Management Information Base
NE	Network Element
NAC	Network Access Card
NMC	Network Management Card
NMS	Network Management System
QOS	Quality of Service
SAP	The largest integrated enterprise software company. Also refers to the software system provided by SAP. SAP provides integrated software for accounting, finance, inventory, supply chain management, customer relations management, supplier

	relations management, HR management and so on. SAP software is used to manage the CommWorks supply chain.
SC	Shelf Controller
SCOPS	Supply Chain Operations
SM	System Manager
SNMP	Simple Network Management Protocol
SPI	Service Provider Interface
TC2000	Total Control 2000
TCP	Transmission Control Protocol
XML	Extensible Markup Language

5 APPLICABLE DOCUMENTS and External Standards Specifications

Title
FAS 30142 SysFS
FAS Agent Sw FS
FAS Sub-Agent FS
Feature ID 56 FFD
FS_FAS_FeatureKey

Generator
FAS Server Detailed Design Document
Secondary Configuration Design Document

6 System Functional Overview

The Feature Activation System is intended to facilitate ease of feature deployment. The primary method of achieving this is based on the use FAS Server [**Figure 1**]. FAS Server requires network feature keys. A network feature key is converted into element feature key using FAS Server. An element feature key is used by the FAS-Agent like System Manager in a network element. The Feature Key Generator can generate network and element feature keys. Element feature keys are also generated by FAS-Server based on the network feature keys.

The fundamental architectural model involves provisioning of the feature identifiers on each blade. This is done by creating a row for each feature identifier in Module Feature Configuration Table in FAS-Subagent MIB. When a row in this table is activated, the blade makes a request of the provisioned feature identifiers from the System Manager (FAS-Agent). System Manager looks into its available pool of feature units for the requested feature identifier (NE Feature Key Status Table). If a feature unit for the requested feature identifier is available at System Manager, it grants permission to the blade to use the requested feature. If the requested feature is not available in System Manager, it contacts the FAS-Server for the requested features. Contingent to availability of requested features on FAS-Server, System Manager gets the requested feature.

System Manager contacts the FAS-Server only if the FAS-Server is provisioned on System Manager. System Manager then grants permission to the blade to use the requested feature. System Manager updates its tables (NE Feature Key Status Table and Blade Feature Key Status Table) to reflect the grant of feature request. It is the blades' responsibility to use the activated features either immediately or after reboot. Blades must request permission from the FAS-Agent before using the features. A save configuration is performed after the blade has been provisioned for the requested features.

All the user interactions between the SNMP management station (e.g. CEM) and FAS-Agent (System Manager) are based on the SNMP protocol. FAS-Agent pulling feature activations from the FAS-Server

are based on TCP protocol. FAS-Subagent requesting permissions from FAS-Agent to use feature activations are based on TCP protocol.

Traps are generated for each failed request of feature from the blade to the System Manager. A trap shall also be generated if SM doesn't get the requested features from a provisioned FAS-Server. System Manager generates all traps.

In the absence of FAS-Server [**Figure 2**] acquisition of element feature keys is done manually. Element feature keys are directly transferred to the FAS-agent. In this case, the Feature Key Generator based on the serial number of the control shelf generates element feature keys.

All feature keys are contained in XML format.

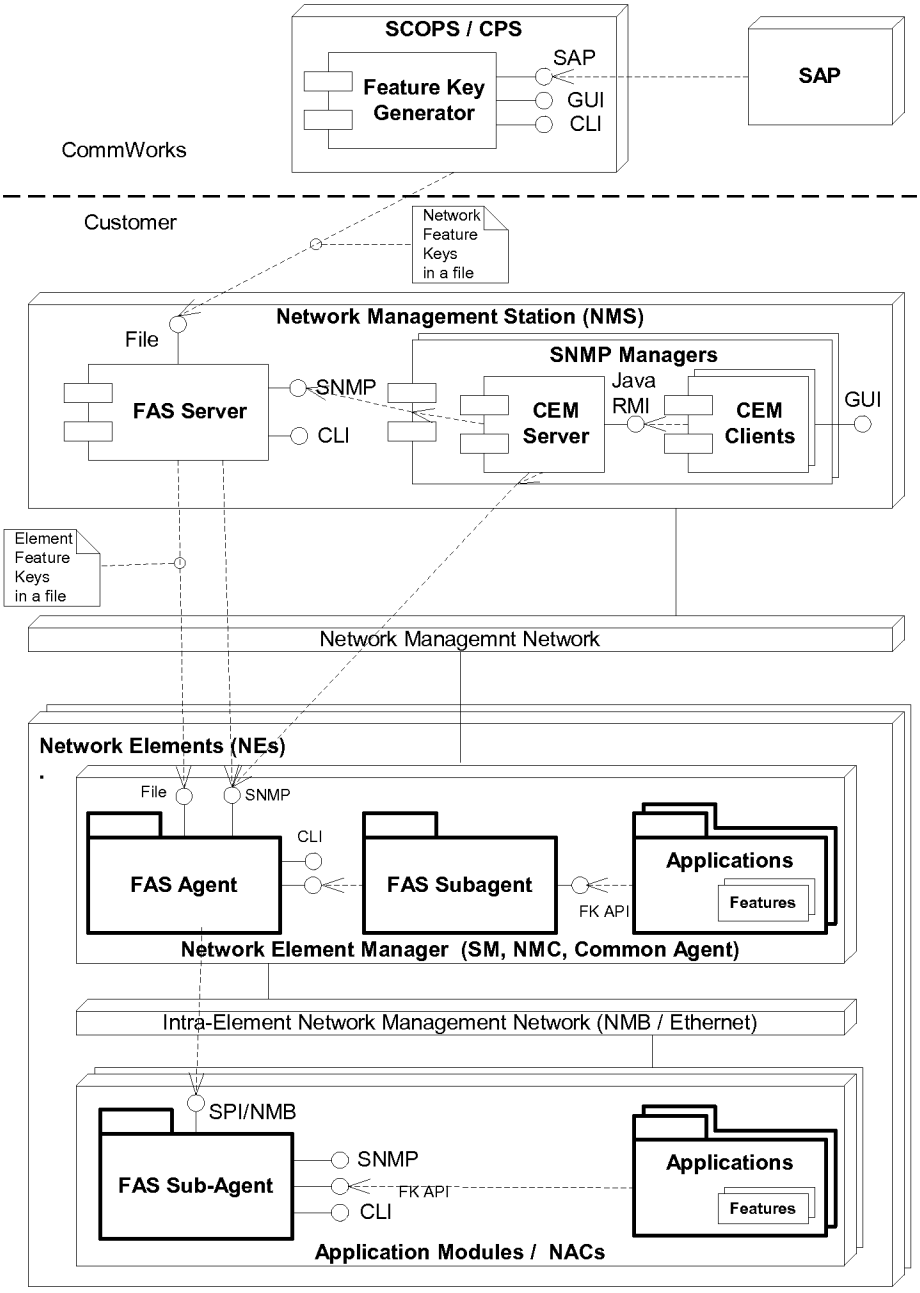


Figure 1 FAS deployment diagram with a FAS Server

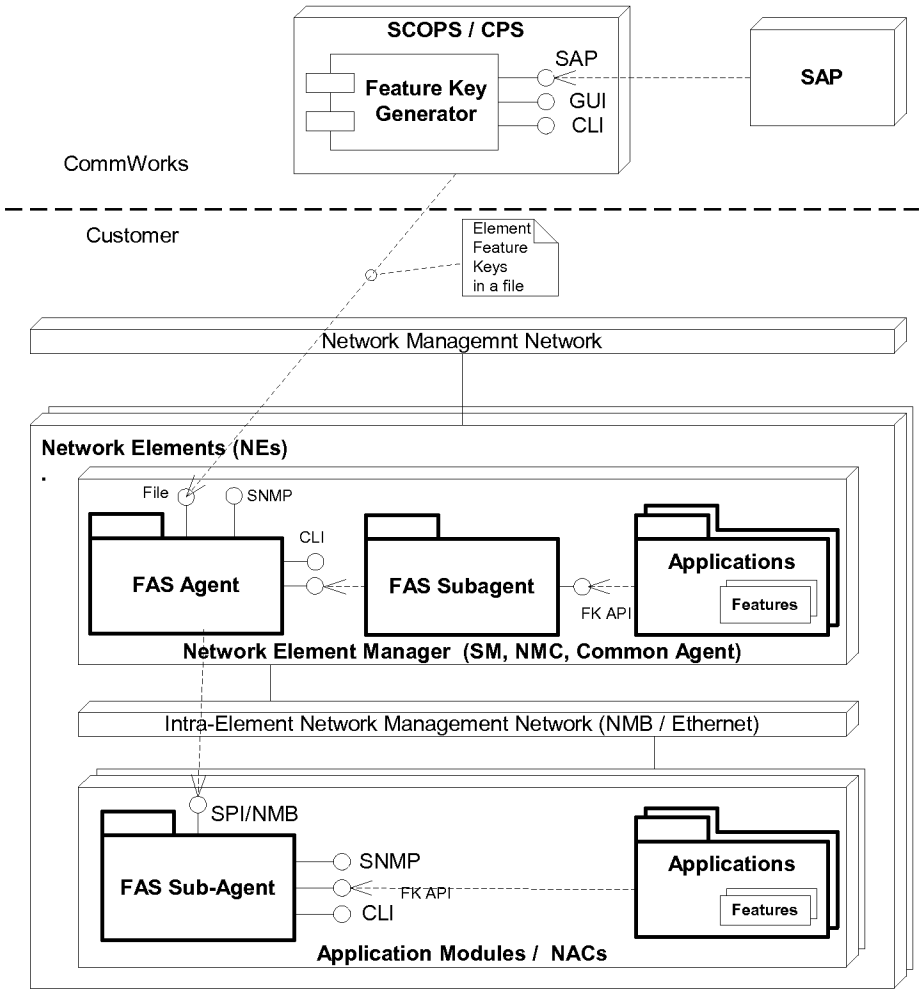


Figure 2 FAS deployment without a FAS Server

7 High-Level Design

FAS-Agent will be a separate daemon process on the System Manager. FAS-Agent communicates with the FAS-SubAgent and FAS-Server using the FASP protocol.

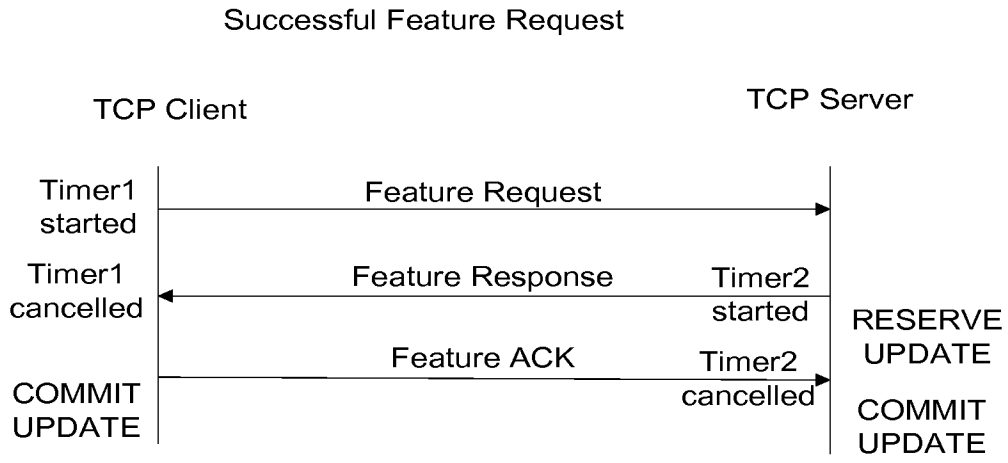
7.1 *Feature Activation System Protocol (FASP)*

Feature Activation System Protocol is a client-server protocol. This protocol shall be used between FAS-Agent and FAS-Server; and between FAS-Subagent and FAS-Agent. The protocol shall be based on TCP. The server shall be running on the FAS-Server. The clients shall be on the FAS-Agents. All transaction

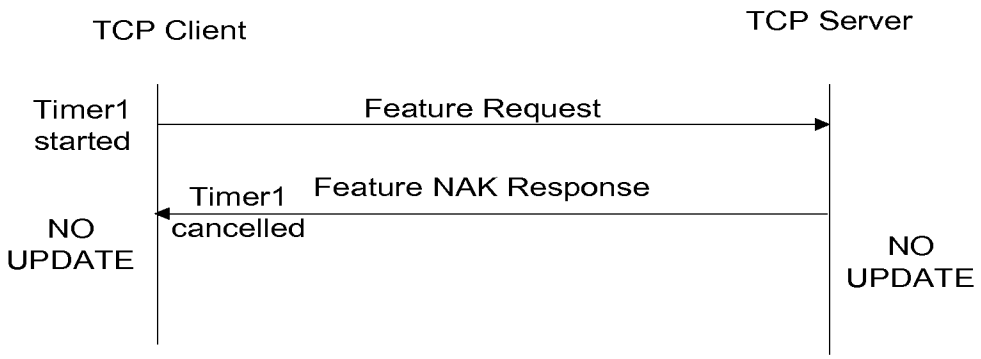
packets shall be authenticated using a symmetric key pair. The feature key transferred in the TCP packet from FAS-Server to the FAS-Agent shall be in XML format. The protocol transactions are below. This protocol shall abide by the following terms:

- Connection initiation shall be done by the TCP-Client.
- Connection can be terminated either by TCP-Client or TCP-Server. In normal conditions, the connection termination shall be initiated by the TCP-Client.
- A connection shall be setup when the first feature request is made. This connection shall not be terminated immediately after a transaction is complete. It shall wait for X duration (to be determined) in anticipation of another feature request from FAS-Subagent.
- Each transaction shall be authenticated between the FAS-Agent and FAS-Server.
- Server must support the same protocol version as requested by the client; or else the request gets rejected indicating the version supported by server. The client may choose to lower its protocol version number in the subsequent requests.
- Detection of an ungraceful connection termination shall rollback the current transaction on both ends.
- After the last transfer of packet over the TCP connection, there should be 2 MSL seconds delay before closing the connection.
- All multi-byte integers shall be encoded using network encoding.

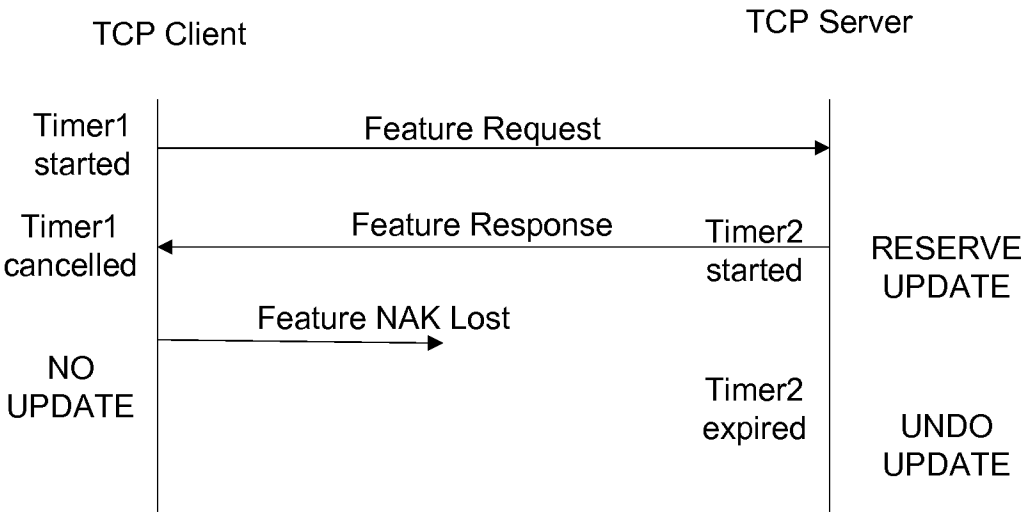
Feature Request

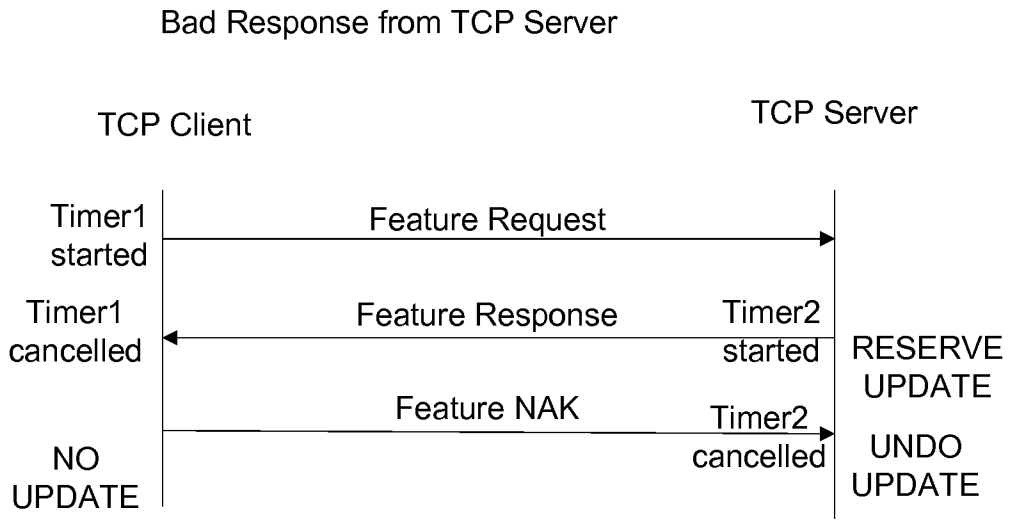


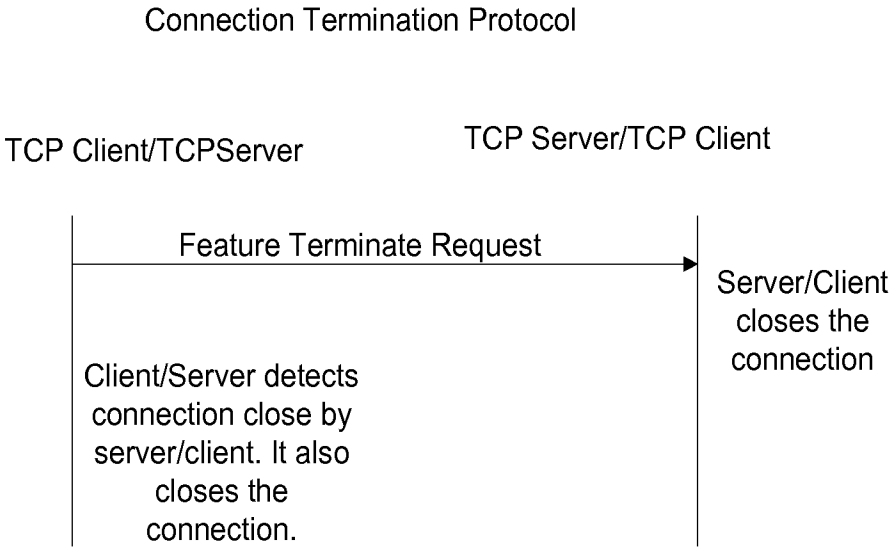
Unsuccessful Feature Request



NAK Lost from TCP Client







7.2 Message Flow Sequence

This section describes the message flow sequences between FAS Subagent, FAS Agent & FAS Server components of FAS system.

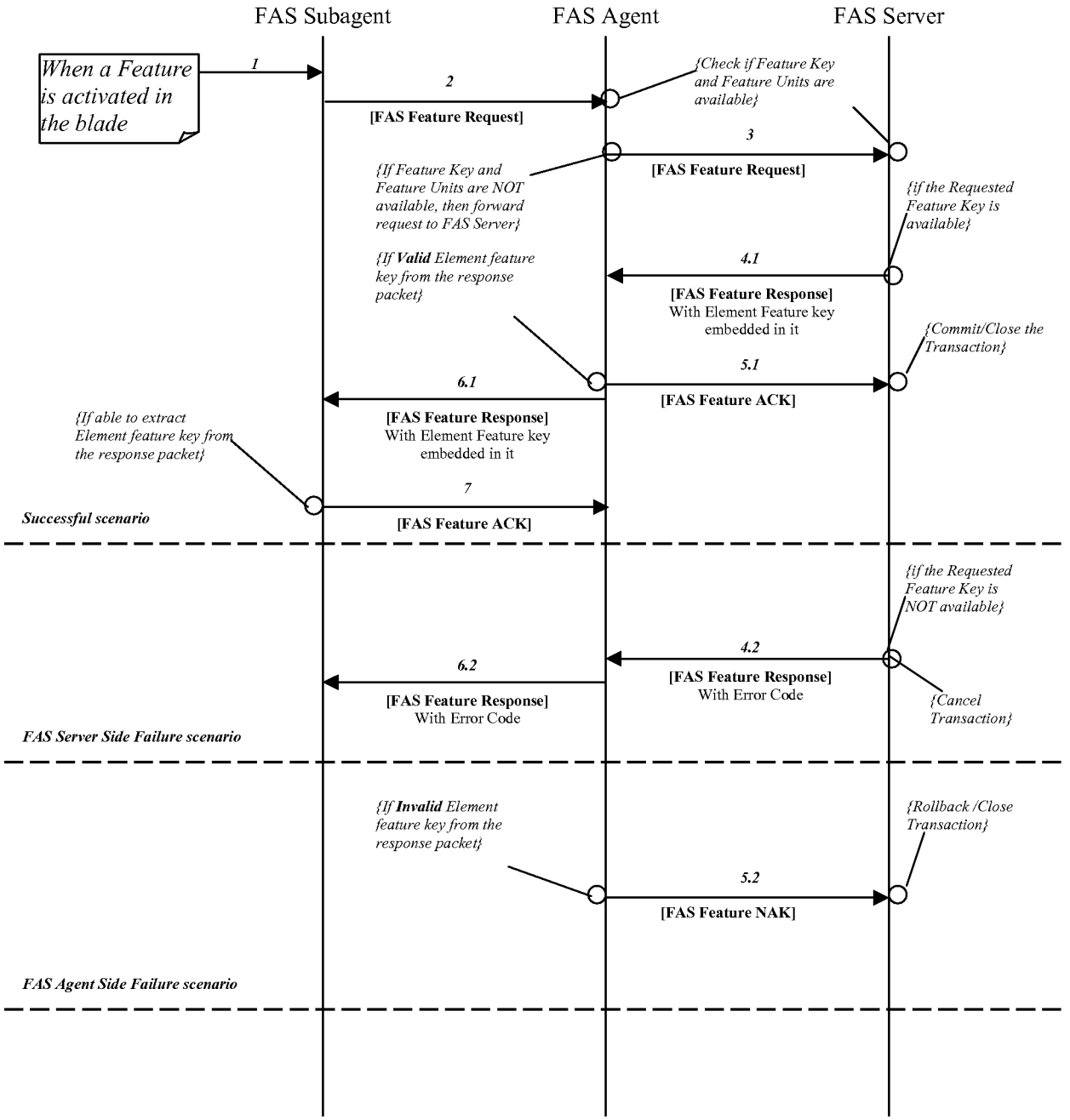


Figure 3 Message Flow Sequence Diagram

8 Detailed Design

8.1 Classification of components

The following list of components has been identified during the design of the architecture. It is defined very briefly with just enough detail to explain how they interact in order to provide the main required functionality. FAS Agent components are:

1. Registration of all management objects with the ACSIL/SPI. The management objects include all the objects from the NE Status Table, Blade Status Table and FAS-Server IP-Address. These addresses are listed in FAS.MIB (Appendix A).

2. Restoration of the NE Status Table, if present, that is saved on the flash of System Manager.

3. Restoration of the Blade Status Table, if present, that is saved on the flash of the System Manager.

4. Launching of a MonitorThread. This thread monitors the incoming connections from the FAS-SubAgents. It takes requests from the head of the ConnectionQueue. The ConnectionQueue is filled up by the ListenerThread. There are a MAX_THREAD (default is 4) ConnectionAccept threads that process the requests from the ConnectionQueue. It takes one request from the ConnectionQueue and starts a new ConnectionAccept Thread. The MonitorThread waits to start a new ConnectionAccept Thread if there are more than MAX_THREAD FAS-SubAgents requesting at the same time.

5. Launching of a ListenerThread. This thread listens for a connection from the FAS-SubAgents. Once a connection request is made by a FAS-SubAgent, the connection is queued in ConnectionQueue. The MonitorThread reads the connection requests from this queue one at a time, and perform the processing.

6. Once the ConnectionThread starts, it processes the request from the FAS-SubAgent.

7. Processing of the requests involves:

- Creating of a transaction object. This object is alive for the duration of the transaction.
- Receiving the packet
- Parsing the header to find out the if the packet is meant for FAS-Agent.
- Create a FasPduRequest Object.
- Parses the body of the packet by loading all the parameters in the FasPduRequest Object.
- Verify each of the tags that need to be present in the packet.
- Check to verify if this is a regular request for feature or a revocation request to revoke a feature. If it is a revocation request, check the Blade Status Table for the presence of the feature to be revoked. If present, create a response packet by creating a FasPduResponse object. Wait for a response from the FAS-SubAgent. If the response from the FAS-SubAgent is an ACK, remove the entry in the Blade Status Table for the requested feature to be revoked for that slot, increment the availableFeatureUnits count in the NE Status Table by 1 and commit the transaction by saving the NE Status Table and Blade Status Table. If the request is a regular request, proceed further.

- Checks to see if the request is present in the Blade Status Table for this request. If yes, then send a response with FEATURE_ALREADY_ASSIGNED response and wait for ACK/NAK response from FAS-SubAgent.

- If the requested feature is not present in the Blade Staus Table, look into the NE Status Table. Check the NE Status Table to find out if there is any available requested feature. If the requested feature is present in the NE Status Table, reduce the AvailableFeatureUnits count in NE Status Table for the requested feature by 1, add an entry in the Blade Status Table for the requested feature for that slot, and send a response to the FAS-SubAgent and wait for ACK/NAK.

- If the feature is not present in the NE Status Table, check if a FAS Server is provisioned in the FAS-Agent. If a FAS Server is provisioned, create a FasPduRequest object and create a request packet. Send this request to the FAS-Server for the requested feature. Wait for a response from FAS-Server. If the response from the FAS-Server is good, then update the NE Status Table. Create an ACK packet by creating a FasPduACK object. Send this ACK to the FAS-Server. Make an entry into the Blade Status Table for the requested feature for that slot. Create a response packet for the FAS-SubAgent by creating a FasPduResponse object. Send this response to the FAS-SubAgent that requested this feature. Wait for an ACK/NAK from the FAS-SubAgent. If an ACK is received, commit the transaction by saving the NE Status Table and Blade Status Table. If a NAK is received from the FAS-SubAgent, then undo the changes in the NE Status Table and Blade Status Table.

- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXX

8. Loading the Element Feature Keys

Element Feature Keys are generated by the Feature Key Generator for a specific Total Control 2000 chassis. These feature keys are stored in an Element Feature Key File. To load this file in the FAS-Agent, an SNMP command is provided. After loading the file, it is reflected in the NE Status Table. The name of the command is **installElementFeatureKeyFile**. Following is the SNMP command that is issued to load the Element Feature Key. The Element Feature Key File must be present in the /tmp/ftp directory on System Manager.

set **mmsmcmdmgmtstation.0** 5 **mmsmcmdfunction.0** installElementFeatureKey **mmsmcmdparam1.0** feature.xml

9. MIB Support

See Section [**MIB Definitions**]

10. CLI Support

See Section [**Configuration and Command Line interfaces**]

11. XML Parser

The XML Parser library from expat-XXXX is being used to parse the XML input coming in from FAS Server in the form of a packet, and XML input coming from the Element Feature Key File when installElementFeatureKeyFile command is issued. Expat is a freely available software and it has been cleared by Legal to use. See Appendix C for the clearance.

12. Save/Restore NE Status Table and Blade Status Table

The NE Status Table will be saved in the /config/shelves_#/slot_#/fasNEStatusTable file. The Blade Status Table will be saved in the /config/shelves_#/slot_#/fasBladeStatusTable. There is no user intervention required to save these tables. Whenever there is any change in these tables due to feature transaction, the most recent table will be saved in the flash of SM. There will be another file called /config/shelves_#/slot_#/sec_config.inf present in the same directory where fasBladeStatusTable and fasNEStatusTable files are present. The content of sec_config.inf file will be the name of the fasBladeStatusTable and fasNEStatusTable file. This is done to be in consistent with the policy files implementation. This will allow backup and restoration of these files outside the Total Control 2000 system.

13. Traps

Generate a trap when FAS-Server is not provisioned and feature request can't be satisfied by FAS-Agent locally.

Generate a trap when the authentication fails between FAS Server and FAS Agent.

Generate a trap for each failed feature request.

Generate a trap if feature not available in FAS-Server.

9 RADIUS Attributes Support

Not Applicable

10 Configuration and Command Line interfaces

CLI# list <dest> fas.blade

Description: The above command lists the FAS Blade Status Table. Only the relevant fields are listed. Relevant fields include indexes (feature id, feature serial number, blade location), feature description, total feature units and enable status.

CLI# list <dest> fas.ne

Description: The above command lists the FAS NE Status Table. Only the relevant fields are listed. Relevant fields include indexes (feature id, feature serial number), feature description, total feature units and available feature units.

CLI# show <dest> fas.ne.<featureID>.<featureSerialNumber> <parameter 1> <parameter 2> ... <parameter n>

Description: The above command gets objects from the FAS NE Status Table. The parameter fields are optional. If the parameter field is not specified, then all the objects corresponding to a particular index is listed. If the parameter field is specified, then only those are shown.

The parameter list include:

-Description

-Key-Version

-Key-Type

-Total-Feature-Units

-Available-Feature-Units

-Start-Date

-End-Date

CLI# show <dest> fas.blade.<featureID>.<featureSerialNumber>.'<shelf#:slot#>' <parameter 1> <parameter 2> ... <parameter n>

Description: The above command gets objects from the FAS Blade Status Table. The parameter fields are optional. If the parameter field is not specified, then all the objects corresponding to a particular index is listed. If the parameter field is specified, then only those are shown.

The parameter list include:

-Description

-Key-Version

-Key-Type

-Total-Feature-Units

-Enable-Status

-Start-Date

-End-Date

CLI# show <dest> fas.server server-address

Description: The above command shows the ip-address of the FAS Server. The "server-address" parameter is optional.

CLI# set <dest> fas server-address=<ip-address of fas server>

Description: The above command is used to configure the ip-address of the FAS Server

CLI# install <dest> fas.featurekey -file-name="<ELEMENT FEATURE KEY FILE>"

Description: The above command is used to install an element feature key file generated by the Feature Key Generator. This command loads the element feature keys present in the specified XML file into the FAS NE Status Table.

11 MIB Definitions

There are two tables and one scalar object implemented to support the FAS MIB (Appendix A). These are as follows:

- **NE Status Table:** This is a double index table. MmFasNEFeatureID and mmFasNEFeatureSerialNumber are the two integer indexes. Other objects in this table are listed below:

mmFasNEFeatureID	Unsigned32,
mmFasNEFeatureSerialNumber	INTEGER,
mmFasNEFeatureDescription	DisplayString,
mmFasNEFeatureKeyVersion	INTEGER,
mmFasNEFeatureKeyType	INTEGER,
mmFasNETotalFeatureUnits	INTEGER,
mmFasNEAvailableFeatureUnits	INTEGER,
mmFasNEFeatureStartDate	DateAndTime,
mmFasNEFeatureEndDate	DateAndTime

- **Blade Status Table:** This is a triple index table. MmFasBladeFeatureID, mmFasBladeFeatureSerialNumber and mmFasBladeLocation are the three indexes. Other objects in this table are listed below:

mmFasBladeFeatureID	Unsigned32,
mmFasBladeFeatureSerialNumber	INTEGER,
mmFasBladeLocation	MmLocation,
mmFasBladeFeatureDescription	DisplayString,
mmFasBladeFeatureKeyVersion	INTEGER,
mmFasBladeFeatureKeyType	INTEGER,
mmFasBladeTotalFeatureUnits	INTEGER,
mmFasBladeEnableStatus	INTEGER,
mmFasBladeFeatureStartDate	DateAndTime,
mmFasBladeFeatureEndDate	DateAndTime

- **FAS Server Address:** This is a configurable object. It is used to provision the ip-address of the FAS Server.

mmFasServerAddress	IpAddress

12 Detailed SYSLOG information

Only SYSLOG warning messages are logged.

13 Debugging Facilities

Compile time debugging is available by enabling the FAS_DEBUG in fas.h file. The debug version of the load can be tested on the simulated System Manager on a Lynx OS PC.

14 Unit Test Plan and Test Cases

14.1 Element Feature Key Loading/Unloading Tests on FAS Agent without FAS Server

Load a single element key from a file on System Manager (FAS Agent)	Verify that key is loaded with no errors. Verify that its characteristics are properly displayed in tables on FAS Agent. Verify that key has no effect on system.
Load multiple element keys from a file on System Manager (FAS Agent)	Verify that keys are loaded with no errors. Verify that their characteristics are properly displayed in tables on FAS Agent.
Attempt to load a key on FAS Agent that already exists on FAS Agent.	Verify that the FAS Agent does not allow key to be loaded again.
Attempt to load a single element key from a file on FAS Agent with an invalid FeatureKeySerialNumber	Verify that it does not succeed and that it fails gracefully.
Attempt to load a single element key from a file on FAS Agent with an invalid featureKeyVersion	Verify that it does not succeed and that it fails gracefully.
Attempt to load a single element key from a file on FAS Agent with an	Verify that it does not succeed and that it fails gracefully.

invalid featureKeyType	
Attempt to load a single element key from a file on FAS Agent with an invalid featureID	Verify that it does not succeed and that it fails gracefully.
Attempt to load a single element key from a file on FAS Agent with an invalid featureDescription	Verify that it does not succeed and that it fails gracefully.
Attempt to load a single element key from a file on FAS Agent with an invalid featureUnitCount	Verify that it does not succeed and that it fails gracefully.
Attempt to load a single element key from a file on FAS Agent with an invalid featureUnitDuration	Verify that it does not succeed and that it fails gracefully.
Attempt to load a single element key from a file on FAS Agent with an invalid customerID	Verify that it does not succeed and that it fails gracefully.
Attempt to load a single element key from a file on FAS Agent with an invalid destinationNodeIDType	Verify that it does not succeed and that it fails gracefully.
Attempt to load a single element key from a file on FAS Agent with an invalid destinationNodeIDType	Verify that it does not succeed and that it fails gracefully.
Attempt to load a single element key from a file on FAS Agent with an invalid signatureSPI	Verify that it does not succeed and that it fails gracefully.
Attempt to load a single element key from a file on FAS Agent with an invalid signature	Verify that it does not succeed and that it fails gracefully.
Attempt to load a single element key from a file on FAS Agent that has had the order of data elements	Verify that it does not succeed and that it fails gracefully.

re-arranged.

Attempt to load invalid multiple element keys from a file on FAS Agent	Verify that it does not succeed and that it fails gracefully.

14.2 FAS-Agent Tests with FAS-Server/FAS-SubAgent

Through CLI on PDSN, activate a feature corresponding to a network key that has been loaded on the FAS server.	Verify it works (tables updated, proper activation status reported, and system behaves as necessary based on feature key)
Through CLI on PDSN, inactivate a feature.	Verify it no longer works, proper activation status reported
Reboot SM for persistent storage	Verify that the Feature Keys are loaded in NE and Blade Status Table

15 Memory And Performance requirement

Same as the System Manager in Total Control 2000 Wireless NextGen System. There is no specific memory and performance requirement for the FAS-Agent.

16 Other Considerations and Contraints

Not Applicable

17 Appendix A – FAS AGENT MIB

```
--**************************************************************************

--*

--*    Copyright XXXXXXX, 3Com Corporation, CommWorks Corporation. All rights reserved.

--*

--*    The information in this software is subject to change without notice
```

```
--*    and should not be construed as a commitment by CommWorks Corporation.
--*
--*    CommWorks Corporation assumes no responsibility for the use or
--*    reliability of its software on equipment which is not supplied by
--*    CommWorks Corportation.
--*
--*    All Rights Reserved. Unpublished rights reserved under the copyright
--*    laws of the United States.
--*
--*    The software contained on this media is proprietary to and embodies
--*    the confidential technology of CommWorks Corportation. Possession, use,
--*    duplication, or dissemination of the software and media is authorixed
--*    only pursuant to a valid writtem license from CommWorks Corporation.
--*
--*    $Revision:  $
--*    $Date:      $
--*
--*    CW-TC2000-FAS-MIB
--*
--*****************************************************************************
CW-TC2000-FAS-MIB DEFINITIONS ::= BEGIN

  IMPORTS

     MODULE-IDENTITY,
     OBJECT-TYPE,
         IpAddress,
     Unsigned32
```

```
    FROM SNMPv2-SMI
      DisplayString,
    DateAndTime
      FROM SNMPv2-TC
    MODULE-COMPLIANCE,
    OBJECT-GROUP
      FROM SNMPv2-CONF
    mmMibsSystemManager,
    MmLocation
     FROM CW-TC2000-DEFINITIONS-MIB;
mmFasMIB MODULE-IDENTITY
    LAST-UPDATED " XXXXXXX "
    ORGANIZATION "Carrier Business Unit"
    CONTACT-INFO "Postal: CommWorks Corporate Offices
                  3800 Golf Road
                  Rolling Meadows, Illinois 60008
                  USA
            Tel:  847-262-5000
            Web:  http://www.commworks.com/"
    DESCRIPTION
      "The MIB module for the System Manager Feature Activation System (FAS) Agent
Management."
    REVISION     " XXXXXXX "
    DESCRIPTION
      "The initial revision of this MIB module."
    ::= { mmMibsSystemManager 7}
mmFasObjects        OBJECT IDENTIFIER ::= { mmFasMIB 1 }
```

```
    mmFasConformance      OBJECT IDENTIFIER ::= { mmFasMIB 2 }

    mmFasNEFeature        OBJECT IDENTIFIER ::= { mmFasObjects 1 }

    mmFasBladeFeature     OBJECT IDENTIFIER ::= { mmFasObjects 2 }

    mmFasAddress          OBJECT IDENTIFIER ::= { mmFasObjects 3 }

--

-- The FAS NE Feature Key Status Table

--


    mmFasNEFeatureKeyStatusTable OBJECT-TYPE

        SYNTAX SEQUENCE OF MmFasNEFeatureKeyStatusEntry

        MAX-ACCESS not-accessible

        STATUS     current

        --REQUIRE  any, manager

        --CONFIG   ignore

        DESCRIPTION

            "This table is used to enumerate the complete features summary

             on a Total Control 2000 system."

        ::= { mmFasNEFeature 1 }

    mmFasNEFeatureKeyStatusEntry OBJECT-TYPE

            SYNTAX     MmFasNEFeatureKeyStatusEntry

            MAX-ACCESS not-accessible

            STATUS     current

            --REQUIRE  any, manager

            --CONFIG   ignore

            DESCRIPTION

                "This entry provides entry for a particular feature identifier and a corresponding
feature serial number."
```

```
        INDEX { mmFasNEFeatureID, mmFasNEFeatureSerialNumber }
        --ROW-CREATE  row-application
        --ROW-DELETE  row-application
        ::= { mmFasNEFeatureKeyStatusTable 1 }
MmFasNEFeatureKeyStatusEntry ::= SEQUENCE {
    mmFasNEFeatureID                Unsigned32,
    mmFasNEFeatureSerialNumber      INTEGER,
    mmFasNEFeatureDescription       DisplayString,
    mmFasNEFeatureKeyVersion        INTEGER,
    mmFasNEFeatureKeyType               INTEGER,
    mmFasNETotalFeatureUnits        INTEGER,
    mmFasNEAvailableFeatureUnits    INTEGER,
    mmFasNEFeatureStartDate             DateAndTime,
    mmFasNEFeatureEndDate               DateAndTime
    }
mmFasNEFeatureID  OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS   not-accessible
  STATUS      current
  --REQUIRE    any, manager
  --CONFIG     ignore
  DESCRIPTION
    "This is the feature identifier of a feature."
  ::= { mmFasNEFeatureKeyStatusEntry 1 }
mmFasNEFeatureSerialNumber  OBJECT-TYPE
  SYNTAX      INTEGER
```

```
    MAX-ACCESS   not-accessible
    STATUS       current
    --REQUIRE    any, manager
    --CONFIG     ignore
    DESCRIPTION
      "This is the feature serial number of a feature. Feature Serial Number
       is assigned either by the Feature Key Generator or the FAS Server."
    ::= { mmFasNEFeatureKeyStatusEntry 2 }
mmFasNEFeatureDescription OBJECT-TYPE
    SYNTAX       DisplayString
    MAX-ACCESS   read-only
    STATUS       current
    --REQUIRE    any, manager
    --CONFIG     ignore
    DESCRIPTION
      "The description of a feature identifier."
    ::= { mmFasNEFeatureKeyStatusEntry 3 }
mmFasNEFeatureKeyVersion OBJECT-TYPE
    SYNTAX       INTEGER
    MAX-ACCESS   read-only
    STATUS       current
    --REQUIRE    any, manager
    --CONFIG     ignore
    DESCRIPTION
      "The version of the feature key."
    ::= { mmFasNEFeatureKeyStatusEntry 4 }
```

```
mmFasNEFeatureKeyType OBJECT-TYPE
   SYNTAX      INTEGER {
                element(1),
                global(2)
               }
   MAX-ACCESS  read-only
   STATUS      current
   --REQUIRE   any, manager
   --CONFIG    ignore
   DESCRIPTION
     "The key type of a feature. Element feature key indicates that the feature is
      meant for a particular Total Control 2000 system. Global feature key indicates
      that the feature is meant for all Total Control system in a network."

   ::= { mmFasNEFeatureKeyStatusEntry 5 }
mmFasNETotalFeatureUnits OBJECT-TYPE
   SYNTAX      INTEGER
   MAX-ACCESS  read-only
   STATUS      current
   --REQUIRE   any, manager
   --CONFIG    ignore
   DESCRIPTION
     "The total number of feature units present in a Total Control 2000
      system for a particular feature identifier and feature serial number."
   ::= { mmFasNEFeatureKeyStatusEntry 6 }
mmFasNEAvailableFeatureUnits OBJECT-TYPE
```

```
    SYNTAX      INTEGER
    MAX-ACCESS  read-only
    STATUS      current
    --REQUIRE   any, manager
    --CONFIG    ignore
    DESCRIPTION
      "The available number of feature units present in Total Control 2000
       system for a particular feature and feature serial number."
    ::= { mmFasNEFeatureKeyStatusEntry 7 }
mmFasNEFeatureStartDate OBJECT-TYPE
    SYNTAX      DateAndTime
    MAX-ACCESS  read-only
    STATUS      current
    --REQUIRE   any, manager
    --CONFIG    ignore
    DESCRIPTION
      "The start date of the feature identifier. If the value of the feature identifier
       is 0, then there is no expiration for the feature."
    ::= { mmFasNEFeatureKeyStatusEntry 8 }
mmFasNEFeatureEndDate OBJECT-TYPE
    SYNTAX      DateAndTime
    MAX-ACCESS  read-only
    STATUS      current
    --REQUIRE   any, manager
    --CONFIG    ignore
    DESCRIPTION
```

```
            "The end date of the feature identifier. If the value of the feature identifier
             is 0, then there is no expiration for the feature."
          ::= { mmFasNEFeatureKeyStatusEntry 9 }
--
-- The FAS Blade Feature Key Status Table
--

   mmFasBladeFeatureKeyStatusTable OBJECT-TYPE
      SYNTAX SEQUENCE OF MmFasBladeFeatureKeyStatusEntry
      MAX-ACCESS not-accessible
      STATUS     current
      --REQUIRE  any, manager
      --CONFIG   ignore
      DESCRIPTION
         "This table is used to enumerate the complete feature assignments
          to each blade on a Total Control 2000 system."
      ::= { mmFasBladeFeature 1 }
   mmFasBladeFeatureKeyStatusEntry OBJECT-TYPE
         SYNTAX     MmFasBladeFeatureKeyStatusEntry
         MAX-ACCESS not-accessible
         STATUS     current
         --REQUIRE  any, manager
         --CONFIG   ignore
         DESCRIPTION
            "This entry provides entry for a particular feature identifier, a feature serial number
and a particular slot."
```

```
                INDEX { mmFasBladeFeatureID, mmFasBladeFeatureSerialNumber,
mmFasBladeLocation }

                --ROW-CREATE  row-application

                --ROW-DELETE  row-application

                ::= { mmFasBladeFeatureKeyStatusTable 1 }

            MmFasBladeFeatureKeyStatusEntry ::= SEQUENCE {
                mmFasBladeFeatureID             Unsigned32,
                mmFasBladeFeatureSerialNumber   INTEGER,
                mmFasBladeLocation              MmLocation,
                mmFasBladeFeatureDescription    DisplayString,
                mmFasBladeFeatureKeyVersion     INTEGER,
                mmFasBladeFeatureKeyType        INTEGER,
                mmFasBladeTotalFeatureUnits     INTEGER,
                mmFasBladeEnableStatus          INTEGER,
                mmFasBladeFeatureStartDate      DateAndTime,
                mmFasBladeFeatureEndDate        DateAndTime
                }

            mmFasBladeFeatureID  OBJECT-TYPE
              SYNTAX      Unsigned32
              MAX-ACCESS   not-accessible
              STATUS      current
              --REQUIRE    any, manager
              --CONFIG     ignore
              DESCRIPTION
                "This is the feature identifier of a feature."
              ::= { mmFasBladeFeatureKeyStatusEntry 1 }

            mmFasBladeFeatureSerialNumber  OBJECT-TYPE
```

```
    SYNTAX      INTEGER
    MAX-ACCESS  not-accessible
    STATUS      current
    --REQUIRE   any, manager
    --CONFIG    ignore
    DESCRIPTION
      "This is the feature serial number of a feature. Feature Serial Number
       is assigned either by the Feature Key Generator or the FAS Server."
    ::= { mmFasBladeFeatureKeyStatusEntry 2 }
mmFasBladeLocation  OBJECT-TYPE
    SYNTAX      MmLocation (SIZE(1..4))
    MAX-ACCESS  not-accessible
    STATUS      current
    --REQUIRE   any, manager
    --CONFIG    ignore
    DESCRIPTION
      "This is the location of the blade on the Total Control 2000 where the
       feature has been assigned."
    ::= { mmFasBladeFeatureKeyStatusEntry 3 }
mmFasBladeFeatureDescription  OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    --REQUIRE   any, manager
    --CONFIG    ignore
    DESCRIPTION
```

```
      "The description of a feature identifier."
   ::= { mmFasBladeFeatureKeyStatusEntry 4 }
mmFasBladeFeatureKeyVersion OBJECT-TYPE
   SYNTAX     INTEGER
   MAX-ACCESS  read-only
   STATUS     current
   --REQUIRE   any, manager
   --CONFIG    ignore
   DESCRIPTION
     "The version of the feature key."
   ::= { mmFasBladeFeatureKeyStatusEntry 5 }
mmFasBladeFeatureKeyType OBJECT-TYPE
   SYNTAX     INTEGER {
                element(1),
                global(2)
               }
   MAX-ACCESS  read-only
   STATUS     current
   --REQUIRE   any, manager
   --CONFIG    ignore
   DESCRIPTION
     "The key type of a feature. Element feature key indicates that the feature is
      meant for a particular Total Control 2000 system. Global feature key indicates
      that the feature is meant for all Total Control system in a network."
   ::= { mmFasBladeFeatureKeyStatusEntry 6 }
mmFasBladeTotalFeatureUnits OBJECT-TYPE
```

```
    SYNTAX      INTEGER
    MAX-ACCESS  read-only
    STATUS      current
    --REQUIRE   any, manager
    --CONFIG    ignore
    DESCRIPTION
      "The total number of feature units present in a Total Control 2000
       system for a particular feature identifier and feature serial number
       for a slot location."
    ::= { mmFasBladeFeatureKeyStatusEntry 7 }
mmFasBladeEnableStatus OBJECT-TYPE
    SYNTAX      INTEGER {
                disable(1),
                enable(2)
                }
    MAX-ACCESS  read-only
    STATUS      current
    --REQUIRE   any, manager
    --CONFIG    ignore
    DESCRIPTION
      "This indicates the status of the feature being used by a blade."
    ::= { mmFasBladeFeatureKeyStatusEntry 8 }
mmFasBladeFeatureStartDate OBJECT-TYPE
    SYNTAX      DateAndTime
    MAX-ACCESS  read-only
    STATUS      current
```

```
        --REQUIRE   any, manager
        --CONFIG    ignore
        DESCRIPTION
          "The start date of the feature identifier. If the value of the feature identifier
           is 0, then there is no expiration for the feature."
        ::= { mmFasBladeFeatureKeyStatusEntry 9 }
      mmFasBladeFeatureEndDate OBJECT-TYPE
        SYNTAX      DateAndTime
        MAX-ACCESS  read-only
        STATUS      current
        --REQUIRE   any, manager
        --CONFIG    ignore
        DESCRIPTION
          "The end date of the feature identifier. If the value of the feature identifier
           is 0, then there is no expiration for the feature."
        ::= { mmFasBladeFeatureKeyStatusEntry 10 }
--
-- FAS Server MIB Group
--
      mmFasServerAddress OBJECT-TYPE
        SYNTAX     IpAddress
        MAX-ACCESS read-write
        STATUS     current
        --REQUIRE   any, manager
        --CONFIG   save
        DESCRIPTION
```

```
            "The IP address of the FAS server to which a FAS Request
             request will be sent."
        ::= { mmFasAddress 1 }
--
-- CONFORMANCE DEFINITIONS
--
    mmFasGroupSet       OBJECT IDENTIFIER ::= { mmFasConformance 1 }
    mmFasComplianceSet  OBJECT IDENTIFIER ::= { mmFasConformance 2 }
    mmFasGroup OBJECT-GROUP
        OBJECTS         {
                        mmFasNEFeatureKeyVersion,
                        mmFasNEFeatureKeyType,
                        mmFasNETotalFeatureUnits,
                        mmFasNEAvailableFeatureUnits,
                        mmFasNEFeatureStartDate,
                        mmFasNEFeatureEndDate,
                        mmFasBladeFeatureKeyVersion,
                        mmFasBladeFeatureKeyType,
                        mmFasBladeEnableStatus,
                        mmFasBladeFeatureStartDate,
                        mmFasBladeFeatureEndDate,
                                mmFasServerAddress
                        }
        STATUS          current
        DESCRIPTION     "All objects in this MIB."
        ::= { mmFasGroupSet 1 }
```

```
    mmFasCompliance MODULE-COMPLIANCE
        STATUS        current
        DESCRIPTION   "The compliance statement for entities which
                      implement CW-TC2000-FAS-MIB."
        MODULE        -- this module
          MANDATORY-GROUPS { mmFasGroup }
        ::= { mmFasComplianceSet 1 }
END -- CW-TC2000-FAS-MIB
```

18 APPENDIX B – FAS Protocol

18.1 Header Information

Protocol Version(1 byte)	Packet Type(1 byte)	Packet Length(2 bytes)

18.1.1 Header Information Description

Field Name	Description
Protocol Version	This indicates the protocol version used. This shall be a simple positive integer. Valid values are from 1 to 255. Initial version is 1.
Packet Type	This is a 1-byte field. The valid values are from 1 to 255.Valid Values of Packet Type:• FAS Feature Request (1)• FAS Feature Response (2) • FAS Feature ACK (3)• FAS Feature NAK (4)• FAS Terminate (5)More packet types may be added if needed.
Packet Length	This is the total length of packet in number of bytes including the

	header information.

18.2 Tag Details

Tag(1 byte)	Length(2 bytes)	Value(variable)

18.2.1 Tags

Tag Name	Tag Numbers	Length (Byte/s)	Tag data Type (Interpreted as)
Packet Subtype	1	1	Unsigned char
Transaction Id	2	4	Integer
Feature Identifier	3	4	Integer
Total Feature Units	4	4	Integer
Packet Response Code	5	4	Integer
Entity ID	6	variable	Character array
Packet Authenticator	7	16	Character array
Auth SPI	8	4	Integer
Element Feature Key Number	9	4	Integer
Element Feature Key	10	4	Character array
Packet Checksum	11	4	Integer
Feature Expiration	12	4	Integer

18.3 Packet Type-Packet Subtype Matrix

Packet Type	Packet Subtype	Protocol Endpoints
FAS Feature Request (1)	1	FAS Agent and FAS Server
FAS Feature Request (1)	2	FAS Subagent and FAS Agent
FAS Feature Response (2)	1	FAS Agent and FAS Server
FAS Feature Response (2)	2	FAS Subagent and FAS Agent
FAS Feature ACK (3)	1	FAS Agent and FAS Server
FAS Feature ACK (3)	2	FAS Subagent and FAS Agent
FAS Feature NAK (4)	1	FAS Agent and FAS Server
FAS Feature NAK (4)		FAS Subagent and FAS Agent

18.4 FAS Feature Request Packet

The feature request packet contains the following fields in the form of tag-length-value.

Packet Subtype	The packet subtype indicates the parameter set contained in the message.
if (packet-subtype == 1) Packet Authencator	Authenticator String
If (packet-subtype == 1) Auth SPI	Specifies which key and algorithm shall be used for authentication

If (packet-subtype == 2) Packet Checksum	Checksum for the entire packet
Transaction Identifier	Unique value to identify packet for a complete transaction. This is generated by FAS-Agent for the Feature Request Packet only. All other kinds of packets in this transaction will use the same packet identifier as in feature request packet.
Entity ID	Serial Number of the Total Control 2000 Control Shelf. This is unique for each Total Control 2000 system.
Feature Identifier	The identifier of the feature being requested.
Total Feature Units	Total number of feature units of a particular feature identifier being requested.

18.5 FAS Feature Response Packet

The feature response packet contains the following fields in the form of tag-length-value.

Packet Subtype	The packet subtype indicates the parameter set contained in the message.
If (packet-subtype == 1) Packet Authenticator	Authenticator String
If (packet-subtype == 1) Auth SPI	Specifies which key and algorithm shall be used for authentication
If (packet-	Checksum of

subtype == 2) Checksum	the entire packet
Transaction Identifier	Unique value to identify packet for a complete transaction. This is the same value as in the feature request packet.
Feature Identifier	The identifier of the feature being requested.
Total Feature Units	Total number of feature units of a particular feature identifier being requested.
Packet Response Code	This contains the error code for the protocol response packet. Valid values include: // Packet Response Codes FAS_NO_ERROR 0 FAS_CHECKSUM_FAILURE 1 FAS_VERSION_MISMATCH 2 FAS_AUTHENTICATION_FAILURE 3 FAS_FEATURE_UNITS_NOT_AVAILABLE 4 FAS_CHECKSUM_NOT_FOUND 5 FAS_AUTHENTICATION_NOT_FOUND 6

	FAS_FEATURE_ID_NOT_FOUND 7 FAS_TOTAL_FEATURE_UNITS_NOT_FOUND 8 FAS_ENTITY_ID_NOT_FOUND 9 FAS_ENTITY_ID_BAD 10 FAS_FEATURE_ALREADY_ASSIGNED 11 FAS_UNABLE_TO_CONNECT_TO_FAS_SERVER 12 FAS_UNABLE_TO_GET_RESPONSE_FROM_FAS_SERVER 13 FAS_UNABLE_TO_RECEIVE_DATA_FROM_FAS_SERVER 14 FAS_INVALID_PACKET_TYPE 15 FAS_INVALID_PROTOCOL_VERSION 16 FAS_UNABLE_TO_ALLOCATE_MEMORY 17 FAS_INVALID_RESPONSE 18 FAS_UNABLE_TO_CREATE_SOCKET 19 FAS_UNABLE_TO_SEND_PACKET

	20 FAS_DATA_FORMAT_ERROR 21 FAS_INVALID_TRANSACTION_ID 22 FAS_FEATURE_NOT_ASSIGNED 23
If (packet-subtype == 1) Feature Key in XML format	Feature Key in XML format It shall be parsed by the FAS-Agent.

18.6 FAS Feature Ack Packet

The feature ack packet contains the following fields in the form of tag-length-value.

Packet Subtype	The packet subtype indicates the parameter set contained in the message.
If (packet-subtype == 1) Packet Authenticator	Authenticator String
If (packet-subtype == 1) Auth SPI	Specifies which key and algorithm shall be used for authentication
If (packet-subtype == 2) Checksum	Checksum for the entire packet
Transaction Identifier	Unique value to identify packet for a complete transaction. This is the same value as in the feature response packet.

18.7 FAS Feature Nak Packet

The feature nak packet contains the following fields in the form of tag-length-value.

Packet Subtype	The packet subtype indicates the parameter set contained in the message.
If (packet-subtype == 1) Packet Authenticator	Authenticator String
If (packet-subtype == 1) Auth SPI	Specifies which key and algorithm shall be used for authentication
If (packet-subtype == 2) Checksum	Checksum for the entire packet
Transaction Identifier	Unique value to identify packet for a complete transaction. This is the same value as in the feature response packet.

18.8 FAS Feature Terminate Packet

The feature terminate packet contains only the header.

19 Appendix C– EXPAT Usage Clearance from Legal

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX